University of Alberta
Department of Electrical and Computer Engineering

Learning hierarchies from knowledge graphs

A thesis status report submitted in fulfillment of the candidacy requirement for the degree of Doctor of Philosophy

Marcin Pietrasik
pietrasi@ualberta.ca

Supervisor: Prof. Marek Reformat

Abstract

This report is submitted in fulfilment of the candidacy requirement for the degree of Doctor of Philosophy in Electrical and Computer Engineering at the University of Alberta. The purpose of the report is to justify "an adequate knowledge of the discipline and of the subject matter relevant to the thesis, and the ability to pursue and complete original research at an advanced level." It starts by providing the motivation behind the work as well as its expected contribution to the knowledge graph and artificial intelligence communities. It continues with a description of the background information pertaining to knowledge graphs, their embeddings, stochastic blockmodelling, and taxonomy induction. The plurality of the report is dedicated to summarizing work that has been completed thus far. Specifically, three projects are discussed, the first two of which deal with the induction of class taxonomies from knowledge graphs as well as the hierarchical clustering of subject entities onto this taxonomy. The third project investigates coarsening as a preprocessing step for knowledge graph embeddings. The report concludes with discussion of avenues for future work as well as a timeline for the remainder of the degree.

Contents

1 Introduction

This candidacy report proposes the research topic for my doctoral work, namely the induction of hierarchical structures from knowledge graphs. It is structured as a summary of three completed and nearly completed projects which will make up a significant share of the final dissertation. In addition to this, it provides a motivation for the work as well as its expected contributions to the scientific community. Furthermore, it briefly outlines the relevant background information and related works, placing it at the intersection of knowledge graphs and artificial intelligence. Finally, it presents a path for future work intended to be completed before the final defense.

1.1 Motivation

Knowledge graphs are data storage structures that rely on principles from graph theory to represent information. Specifically, facts are stored as triples which bring together two entities via a predicate. In a graphical context, these entities are analogous to nodes, and the relations between them are analogous to edges.

In recent years, knowledge graphs have garnered widespread attention as a medium for storing data on the web. Public knowledge graphs such as DBpedia [1], YAGO [2], and WikiData [3] are all underpinned by large-scale knowledge graphs containing upwards of one billion triples each. These knowledge graphs find uses in personal, academic, and commercial domains and are ubiquitous in the research fields of the Semantic Web, artificial intelligence, and computer science broadly. Furthermore, private companies are known to use proprietary knowledge graphs as a component of their data stores. Google, for instance, uses a knowledge graph derived from Freebase [4] to enhance their search engine results by providing infoboxes which summarize facts about a user's query [5].

In this report, the titular *Learning Hierarchies from Knowledge Graphs* refers to a broad concept that encompasses distinct tasks related by their induction of hierarchical relations between knowledge graph entities. The clearest example of a knowledge graph hierarchy is the class taxonomy which organizes a knowledge graph's classes through superclass-subclass relations. The task of inducing such a taxonomy merely amounts to learning how the classes are organized hierarchically in the knowledge graph. Similarly, hierarchical clustering of knowledge graphs allows not only to discover which entities are semantically similar as per the clustering but also how entities relate to one another hierarchically. Perhaps less obviously, knowledge graph coarsening may be viewed as a form of hierarchy learning. This task involves collapsing knowledge graph entities with one another based on some – generally structural – similarity. By iteratively coarsening a knowledge graph, a chain of progressively smaller knowledge graphs is built and a hierarchy emerges. These three tasks are further elucidated upon in this report and novel solutions for them are proposed.

The motivating factors behind learning knowledge graph hierarchies are various. Perhaps the simplest is that hierarchical structures organize data in a way that is highly intuitive and interpretable to humans. For instance, a hierarchical clustering of knowledge graph entities makes it apparent which entities constitute the broadest concepts in the knowledge graph and how they relate to their descendants. Similarly, a taxonomy of classes reveals implicit relations between entities through its transitive properties. Put plainly, hierarchies induced from knowledge graphs are useful because they are easy to understand. Indeed, the most widely used knowledge bases such as DBpedia, YAGO, and WikiData are organized by hierarchical structures, namely trees and directed acyclic graphs. That is to say, knowledge graphs are hierarchical at their core. Furthermore, hierarchies are used as components of larger systems to solve common tasks related to knowledge graphs. For instance, hierarchies are used in learning knowledge graph embeddings, both explicitly as an input feature of the model [6] and implicitly as a byproduct of the embedding process [7]. As embedding is one of the most common problems in the knowledge graph community, learning accurate hierarchies is valuable.

1.2 Expected Contributions

The expected contributions of this doctoral work are to develop novel methods and models for learning hierarchies from knowledge graphs using artificial intelligence. Although broad in scope and consisting of several independent projects, the work seeks to advance both hierarchy induction as well as application of hierarchies to common tasks and problems in the knowledge graph community. For instance, one project induces a class taxonomy from knowledge graphs using a novel algorithm based on class frequencies and co-occurrences. Another project aims to show that generating embedddings on a hierarchically coarsened graph often yields higher quality results with reduced computational demand. These projects are unified in that the learning of hierarchies, whether explicit or implicit, constitutes a significant part of each work. Finally, a significant portion of time has and will continue to focus on marrying probabilistic graphical models with knowledge graphs. Specifically, utilizing stochastic blockmodels for modelling knowledge graphs is an understudied area that this work seeks to contribute to. By developing stochastic blockmodels specifically suited to the domain of knowledge graphs, the foundations will be set for further adoption of such models by the wider knowledge graph community.

2 Background

2.1 Knowledge Graphs

A knowledge graph, $\mathcal{K}$, is a repository of information structured as a collection of triples where each triple relates a subject, s, to an object, o, through a predicate, p. More formally, $\mathcal{K} = \{\langle s, p, o\rangle \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}\}$ where $\langle s, p, o\rangle$ is a triple, $\mathcal{E}$ is the set of entities in $\mathcal{K}$, and $\mathcal{P}$ is the set of predicates in $\mathcal{K}$. $\mathcal{K}$ can therefore be viewed as a directed graph with nodes representing entities and edges representing predicates. Each triple in a knowledge graph describes one piece of information, or a fact. For instance, ⟨*dbr:Henry_Ford, dbo:Occupation, dbr:Engineer*⟩ relates the subject *dbr:Henry_Ford* to the object *dbr:Engineer* through the predicate *dbo:Occupation* and states that, in plain English, Henry Ford's occupation is an engineer. When put together, triples form a graph as shown in Figure 1 which visualizes a subset of DBpedia triples describing Henry Ford and his wife Clara. Notice that knowledge graphs allow for cycles and self-relations as shown through the two $dbo:spouse$ and $rdfs:seeAlso$ relations, respectively.

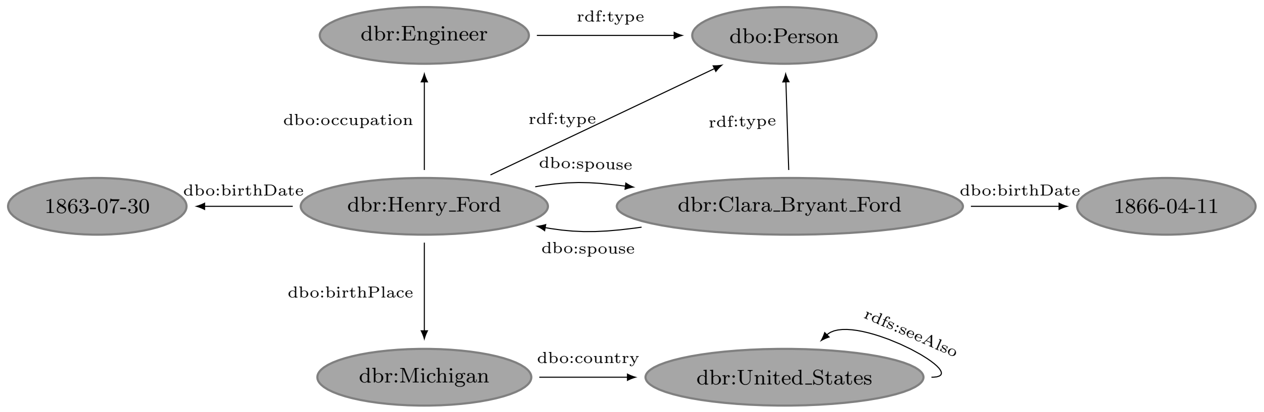

Figure 1: Excerpt of DBpedia triples describing Henry Ford and his wife Clara.

Predicate-object pairs, $\langle p, o \rangle$, can be thought of as tags that describe a subject. In this view, each entity that takes on the role of subject, s_i, is annotated by tags, $t_j \in \mathcal{A}_i$, where $\mathcal{A}_i$ is the set of tags that annotate s_i. The set of subjects is defined as, $s_i \in \mathcal{S}$, such that it is a subset of all entities, $\mathcal{S} \subseteq \mathcal{E}$. Tags are defined as predicate-objects pairs, $t := \langle p, o \rangle$, and belong to the set of all tags – called the vocabulary – denoted as $\mathcal{V}$ such that $\mathcal{A}_i \subseteq \mathcal{V}$. For a concrete example of this notation consider Figure 1, wherein the entity *dbr:Clara_Bryant_Ford* is annotated by the tags ⟨*rdf:type,dbo:Person*⟩, ⟨*dbo:birthDate,1866-04-11*⟩, and ⟨*dbo:spouse, dbr:Henry_Ford*⟩. In this view, the knowledge graph $\mathcal{K}$ may be represented as the set of subject-tag tuples $\mathcal{K} = \{\langle s, t \rangle \in \mathcal{S} \times \mathcal{V}\}$, where $\langle s, t \rangle$ is the tuple that relates subject s with tag t. This notation is referred to as the tuple structure of a knowledge graph for the remainder of the report.

2.1.1 Resource Description Framework

The prefixes *dbr*, *dbo*, *rdf*, and *rdfs* in Figure 1 are artefacts of the notation used as part of the Resource Description Framework (RDF)[1]. RDF refers to a set of standards introduced by the World Wide Web Consortium [2] for representing and exchanging data on the web. It uses the triple structure as its foundation and includes notations and formats for serializing triples. For instance, in the N-Triples format, the three leftmost triples in Figure 1 are written as:

```
@prefix dbo:  <http://dbpedia.org/ontology/> .
@prefix dbr:  <http://dbpedia.org/resource/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema> .
dbr:Henry_Ford dbo:birthDate "1863-07-30"^^xsd:date .
dbr:Henry_Ford dbo:birthPlace dbr:Michigan .
dbr:Henry_Ford dbo:occupation dbr:Engineer .
```

Where the first three lines define Universal Resource Identifiers (URIs) as prefixes before

[1]https://www.w3.org/RDF/
[2]https://www.w3.org/

using them to define the triples themselves. Using such prefixes decreases the size of storing triples and enhances readability. This type of format may be queried using languages such as SPARQL [8] and Cypher [9].

2.1.2 Ontologies

Ontologies are often used in conjunction with knowledge graphs to provide an axiomatic foundation on which knowledge graphs are built. In this view, an ontology may be seen as a vocabulary and a rule book that provides semantics to a knowledge graph and governs how the information contained within it is represented and how it can be reasoned with. Perhaps the most salient feature of ontologies is their definition of knowledge graph classes as well as rules for how they are organized and interact with one another. In adding a layer which conceptually sits on top of a knowledge graph, ontologies both enrich the knowledge graph by allowing it to be reasoned with in more complex ways and constrain it by introducing boundaries for what is and isn't possible. Consider, for instance, the classes *Parent* and *Child* and the predicate *hasChild*. An ontology can provide structure to the predicate by defining its domain and range. Specifically, it can state that the subject related by the predicate *hasChild* has to be of class *Parent* and the object of class *Child*. In the Web Ontology Language (OWL) – a widely used collection of languages for defining ontologies – this constraint may look as follows:

```
<owl:ObjectProperty rdf:ID="hasChild">
 <rdfs:domain rdf:resource="Parent"/>
 <rdfs:range rdf:resource="Child"/>
</owl:ObjectProperty>
```

In addition to the aforementioned constraints on domain and range, ontologies allow for defining subsumption, transitivity, symmetricity, cardinality, equivalence, set operations, and enumeration amongst many more. Of particular importance to this report is the use of subsumption rules, which define superclass-subclass relationships between classes. This allows for building of class taxonomies to organize concepts hierarchically. Consider, for instance, the following definitions:

```
<owl:Class rdf:ID="Dog">
 <rdfs:subClassOf rdf:resource="Mammal"/></owl:Class>
<owl:Class rdf:ID="Cat">
 <rdfs:subClassOf rdf:resource="Mammal"/></owl:Class>
<owl:Class rdf:ID="Mammal">
 <rdfs:subClassOf rdf:resource="Animal"/></owl:Class>
<owl:Class rdf:ID="Turtle">
 <rdfs:subClassOf rdf:resource="Reptile"/></owl:Class>
<owl:Class rdf:ID="Reptile">
 <rdfs:subClassOf rdf:resource="Animal"/></owl:Class>
```

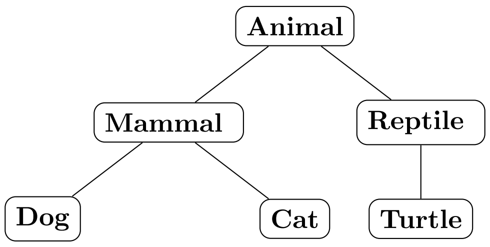

Figure 2: Toy example of class taxonomy describing the relations between different animal classes.

These rules form a taxonomy between five animal classes as shown in Figure 2. Building such hierarchies enriches a knowledge graph as it allows for reasoning that instances of the *Dog* class are also instances of *Mammal* and *Animal*. Furthermore, it provides a conceptual basis for how classes are related to one another. *Dog* and *Cat*, for example, are more closely related conceptually than *Dog* and *Turtle*.

2.2 Embeddings

In mathematics, embeddings are defined as an injective mapping between two objects which preserves properties in the domain. Formally, an embedding function, f, maps object x to its embedding y as follows:

$$f : x \mapsto y \tag{1}$$

A more granular definition of property preservation as well as the constraints and conditions of the mapping vary depending on the types of objects being embedded and the branch of mathematics being studied.

In this report, and in artificial intelligence more generally, embeddings are used in a narrower context. Specifically, embeddings are continuous vector representations of entities, mapped from their original, or ambient, space. The vector space that is formed by the set of embeddings is called the embedding space and is usually of a lower dimension than the ambient space. Generally, embeddings rely on the intuition that entities which are embedded close to one another in the embedding space share similar semantics or properties which are intended to be preserved in the embedding process. Such representations are desired as they are both intuitive and highly operable by machines. For instance, embeddings allow for the calculation of distance between any two entities, revealing potentially useful information. Furthermore, if the ambient space is discrete, the embedding process provides a representation which opens the door to the range of tools and methods in artificial intelligence which operate on continuous inputs.

2.2.1 Graph Embeddings

Graph embeddings are continuous vector representations of a graph's entities and, in some cases, relations. Thus, the task of generating graph embeddings involves finding a function, f, which maps each entity to the embedding space. Defined formally, $f : \mathcal{E} \mapsto \mathbb{R}^{|\mathcal{E}| \times d}$ where d is the dimensionality of the embedding space such that $d << |\mathcal{E}|$. The obtained embeddings are subsequently used to solve downstream tasks such as link prediction and entity classification.

Research in graph representation, including graph embeddings, has a long history rooted in mathematics with early methods discussed in [10]. More recently, deep learning has been leveraged for graph embeddings to achieve state-of-the-art results. For instance, DeepWalk [11], Large-scale Information Network Embedding [12], and node2vec [13] sample random walks on a graph and treat them as input words to the skip-gram language model [14]. The intuition behind this approach is that entities which are sampled in the same random walks are more similar semantically and should have similar embeddings. Another class of deep approaches, Graph Convolution Networks (GCN) [15, 16], utilize the convolution operator to learn neighbourhood information for graph entities. Extensions and derivatives of GCNs are ample; for a comprehensive discussion of these methods readers are referred to [17]. Autoencoders use neural networks to reduce an input to a latent embedding before reconstructing the embedding back to the original input. This approach has shown success in generating graph embeddings in [18] and[19].

Knowledge graphs present structural traits which render the aforementioned methods ill-suited for their embedding. Specifically, knowledge graphs are directed and labelled in their relations between entities. In light of this, much research has been devoted to developing methods which account for these complexities. For instance, RDF2Vec [20] uses breadth-first graph walks on the skip-gram model to generate embeddings. GCNs have been extended to knowledge graphs with the Relational Graph Convolution Network (R-GCN) [21] which aggregates predicate specific convolutions of the original model. ConvE [22] also leverages the convolution operator in a neural framework by stacking embeddings as a martix and convolving them in two dimensions. Translation based methods such as TransE [23] apply the intuition that subject embeddings should be near object embeddings when translated by valid corresponding predicates. Factorization models such as RESCAL [24] and DistMult [25] learn embeddings by factorizing the knowledge graph adjacency tensor into the product of entity embeddings and relation specific translation matrices. Deep reinforcement learning has also shown promise in this domain with MINERVA [26] which learns knowledge graph paths to find the correct entity in incomplete triples. A recent and comprehensive discussion of knowledge graphs embedding methods can be found in [27].

2.3 Stochastic Blockmodels

Stochastic blockmodels belong to a class of probabilistic methods which seek to learn the underlying probability distributions that govern a graph. These methods decompose a graph into probability distributions – called blocks – which represent the different components that are assumed to make up the graph. When these blocks are sampled from, the graph is generated. Thus, the learning process is to infer the parameters of the underlying probability distributions. Most stochastic blockmodels also assign graph entities to communities as part of their generative process. This has the effect of implicitly clustering entities before modelling the relations between them.

Given a graph in the form of an adjacency matrix, $g_{i,j} \in \mathbf{G}$, the stochastic blockmodel's latent parameters, $\mathbf{A}$ and $\mathbf{B}$, and their corresponding hyperparameters, α and β, the joint distribution of the model is as follows:

$$\mathbb{P}(\mathbf{G}, \mathbf{A}, \mathbf{B}|\alpha, \beta) = \prod_{g_{i,j} \in \mathbf{G}} \mathbb{P}(g_{i,j}|\mathbf{A}_{i,j}, \mathbf{B}_{i,j}, \alpha, \beta)\mathbb{P}(\mathbf{A}_{i,j}|\mathbf{B}_{i,j}, \alpha)\mathbb{P}(\mathbf{B}_{i,j}|\beta) \tag{2}$$

Where $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$ indicate the latent parameters in $\mathbf{A}$ and $\mathbf{B}$ associated with sampling $g_{i,j}$. In most cases, the solution is intractable for exact inference and must be approximated using an inference scheme.

Perhaps the most common inference scheme used in stochastic blockmodelling is Gibbs sampling, which approximates a joint distribution by iteratively sampling from its variables' conditional distributions. Continuing the example above, to infer the blockmodel's parameters for $g_{i,j}$, namely $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$, inference is performed on their posterior distributions, $\mathbb{P}(\mathbf{A}_{i,j}|\mathbf{G}, \mathbf{B}, \alpha)$ and $\mathbb{P}(\mathbf{B}_{i,j}|\mathbf{G}, \mathbf{A}, \beta)$, respectively. By Bayes' theorem, the posterior is proportional to the product of the likelihood and the prior, and can therefore be expressed as follows:

$$\mathbb{P}(\mathbf{A}_{i,j}|\mathbf{G}, \mathbf{B}, \alpha) \propto \mathbb{P}(\mathbf{G}|\mathbf{A}_{i,j}, \mathbf{B})\mathbb{P}(\mathbf{A}_{i,j}|\alpha) \tag{3}$$
$$\mathbb{P}(\mathbf{B}_{i,j}|\mathbf{G}, \mathbf{A}, \beta) \propto \mathbb{P}(\mathbf{G}|\mathbf{B}_{i,j}, \mathbf{A})\mathbb{P}(\mathbf{B}_{i,j}|\beta) \tag{4}$$

Where $\mathbb{P}(\mathbf{G}|\mathbf{A}_{i,j}, \mathbf{B})$ and $\mathbb{P}(\mathbf{G}|\mathbf{B}_{i,j}, \mathbf{A})$ are the likelihoods, and $\mathbb{P}(\mathbf{A}_{i,j}|\alpha)$ and $\mathbb{P}(\mathbf{B}_{i,j}|\beta)$ are the priors of $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$, respectively. The likelihood may be understood as the probability of observing the data given the parameters. The prior represents the assumptions about the parameter before any data is taken into account. Gibbs sampling samples the parameters' posteriors iteratively for a predetermined number of iterations. To highlight this, the superscript i is added to denote the value of a parameter at iteration i. The Gibbs sampling process may be summarized as follows:

1. Initialize $\mathbf{A}^0_{i,j}$ and $\mathbf{B}^0_{i,j}$ for each $g_{ij} \in \mathbf{G}$
2. For iteration i in $1, 2, ..., max\_iters$
   (a) Sample $\mathbf{A}^i_{i,j} \sim \mathbb{P}(\mathbf{A}^i_{i,j}|\mathbf{G}, \mathbf{B}^{i-1}, \alpha)$ for each $g_{ij} \in \mathbf{G}$ using Equation 3
   (b) Sample $\mathbf{B}^i_{i,j} \sim \mathbb{P}(\mathbf{B}^i_{i,j}|\mathbf{G}, \mathbf{A}^{i-1}, \alpha)$ for each $g_{ij} \in \mathbf{G}$ using Equation 4

This iterative sampling creates a Markov chain of samples wherein its stationary distribution approximates the joint distribution of the model. In this view, Gibbs sampling may be seen as the learning process of the model since successive samples yield parameters which are more likely to generate $\mathbf{G}$.

The seminal work in this area is the Stochastic Blockmodel (SBM) [28] which uses a latent community approach such that each entity in the graph is assigned to one of a fixed number of communities. Entity relations are then modeled as those of their communities via the community interactions matrix, which models the degree of interaction between communities. The fixed community structure of the SBM was extended to the infinite case in the Infinite Relational Model [29]. This extension allows for modelling multilayer graphs and was mainly used in social network analysis. Perhaps the most studied and extended blockmodel today is the Mixed Membership Stochastic Blockmodel (MMSB) [30]. As its name suggests, the

MMSB allows each entity to have a mixed membership degree by modelling the probability distribution of entities belonging to communities.

Multilayer blockmodels are blockmodels that account for graphs where relations between entities exist on different relation types. This formulation is similar to knowledge graphs wherein entities are related to one another on different predicate types. One trivial approach to multilayer modelling is to apply a blockmodel independently on each relation in the graph as was done in [31] and [32]. The drawback of this approach, however, is that community interactions and memberships will differ with respect to the relation. Drawing conclusions about community structure is therefore difficult. To overcome this, the community interactions can be held constant (shared) across relations. Such an approach allows for interpretability in terms of how network communities interact with one another, but understanding the nature of entity membership to these communities is still hindered. Recently, multilayer extensions of the SBM were proposed in [33] and [34] which allow for modelling the pairwise interactions of entities across relations. Similarly, the MMSB was extended to multilayer graphs in [35] by modelling for shared communities whose interactions are modified depending on their relation.

2.4 Taxonomy Induction

Taxonomy induction refers to the task of learning subsumption axioms for classes in a knowledge graph, thereby inducing a taxonomy. To this end, Statistical Schema Induction [36] uses association rule mining on a knowledge graph's transaction table to generate ontology axioms. Each row in the transaction table corresponds to a subject in the graph along with the classes it belongs to. Implication patterns which are consistent with the table are mined from this table to create candidate ontology axioms. The candidate axioms are then sorted in terms of descending certainty values and added greedily to the ontology only if they are logically coherent with axioms added before them. [37] proposes a method using hierarchical clustering on a decomposed representation of the knowledge graph. Specifically, it extends RESCAL [24], a method for factorizing a three-way tensor, to better handle sparse large-scale data and applies OPTICS [38], a density based hierarchical clustering algorithm. TIEmb [39] relies on entity and text embeddings with the intuition that entities of a subclass will be embedded within their parent class's embeddings. Thus if you calculate the centroid for each class's embeddings, you can infer its subclasses as those whose centroid falls within a certain radius. For instance, the class centroids of *Mammals* and *Reptiles* will fall inside the radius of *Animals* although the converse is not true since *Mammals* and *Reptiles* are more specific classes and are expected to have a smaller radius.

3 Work to Date: A Simple Method for Inducing Class Taxonomies in Knowledge Graphs

This section summarizes the work presented in the papers: *A Simple Method for Inducing Class Taxonomies in Knowledge Graphs* [40] which was published in the proceedings of the 17$^{\text{th}}$ European Semantic Web Conference; and *Path Based Hierarchical Clustering on Knowledge Graphs* [41] which exists as a preprint on arXiv.

3.1 Motivation

One of the core components of an ontology is the class taxonomy: a set of subsumption axioms between the type classes that may exists in the knowledge graph. When put together, the subsumption axioms form a hierarchy of classes where general concepts appear at the top and their subconcepts appear as their descendants. A challenge that arises when working with large knowledge graphs is that of class taxonomy construction. Manual construction is time consuming and requires curators knowledgeable in the area. DBpedia, for instance, relies on its community to curate its class taxonomy. Similarly, YAGO relies on a combination of information from Wikipedia[3] and WordNet[4], both of which are manually selected and organized. On the other hand, automated methods are not able to induce class taxonomies of the quality necessary to reliably apply to complex knowledge graphs. Furthermore, they oftentimes rely on external information which may itself be manually curated or may only be applicable to knowledge graphs in a particular domain. With this in mind, the impetus for automatically inducing class taxonomies of high quality from large-scale knowledge graphs becomes apparent.

3.2 Problem Description

An ontology's class taxonomy is usually formalized as a set of class subsumption axioms. These subsumption axioms may be thought of as is-a relations between classes. For instance, in the DBpedia class hierarchy, the subsumption axioms $\{$*dbo:Person* $\rightarrow$ *dbo:Artist*$\}$ and $\{$*dbo:Artist* $\rightarrow$ *dbo:Painter*$\}$ imply that *dbo:Painter* is a *dbo:Artist* and that *dbo:Artist* is a *dbo:Person*. Furthermore, since class subsumption axioms are transitive, *dbo:Painter* is a *dbo:Person*. This taxonomy oftentimes takes the form of a rooted tree with a root class of which all other classes are considered logical descendants of.

The problem of class taxonomy induction from knowledge graphs involves generating subsumption axioms from triples to build the class taxonomy. It's noticed that in most knowledge graphs, subjects are related to their class type by one predicate. This has the effect of reducing the knowledge graph's class identifying triples to a single dimension. The property can be exploited in the tuple structure, since all class identifying predicates are the same, they can be ignored without loss of information. For instance, in DBpedia the predicate which relates subjects to their class is *rdf:type*. Thus, when compiling a dataset of class identifying tuples, the tag $\langle$*rdf:type,dbo:Country*$\rangle$ and *dbo:Country* can be treated as equivalent. Therefore, the tuple $\langle$*dbr:Canada*, *dbo:Country*$\rangle$ preserves all information required to induce a class taxonomy. This can be utilized by tag hierarchy induction methods which take documents and their tags as input.

3.3 Approach

The proposed method uses class frequencies and co-occurrences to calculate similarity between tags. This approach, inspired by a method proposed by Schmitz in [42], relies on the intuition that subclasses will co-occur in subjects with their superclasses more often than

[3]https://www.wikipedia.org/

[4]https://wordnet.princeton.edu/

with classes they are not logical descendants of. Unlike Schmitz's method which uses this assumption to generate candidate subsumption axioms, the proposed method uses similarity to choose a parent tag which already exists in the taxonomy. In this step, which draws inspiration from Heymann and Garcia-Molina [43], tags are greedily added to the taxonomy in order of decreasing generality. Thus, subsumption axioms induced by this method have to abide by the following rules:

- The parent tag has a higher generality than the child tag.
- The parent tag is the tag with the highest similarity to the child tag from the tags that exist in the taxonomy when the child tag is being added.

The induced class taxonomy can be populated with subject entities, which has the effect of hierarchically clustering them. The process for this is to merely find the class in the hierarchy to which the subject belongs to and assign it to that class. Each class can then be treated as a cluster and its constituent subject entities as cluster elements. The result of this is a hierarchical structure of clusters annotated by tags and with strong subsumption properties.

As previously mentioned, the proposed approach leverages the tuple structure of a knowledge graph to induce a class taxonomy in the form of a rooted tree. As such, the first step is data preprocessing wherein all of a knowledge graph's class identifying triples are converted to tuple structure.

3.3.1 Class Taxonomy Induction Procedure

Before describing the taxonomy induction procedure for the proposed method, the following measures are calculated on the knowledge graph as required input for the algorithm:

- The number of subject entities annotated by tag t_a is denoted as D_a.
- The number of subject entities annotated by both tags t_a and t_b is denoted as $\mathrm{D}_{a,b}$. Note that this measure is symmetrical, i.e. $\mathrm{D}_{a,b} = \mathrm{D}_{b,a}$.
- The generality of tag t_a, denoted as G_a, measures how general the concept described by the tag is and how high it belongs in the taxonomy. The generality is defined as:

$$\mathrm{G}_a = \sum_{t_b \in \mathcal{V}_{-t_a}} \frac{\mathrm{D}_{a,b}}{\mathrm{D}_b} \tag{5}$$

  Where $\mathcal{V}_{-t_a}$ is the set of all tags excluding tag t_a.

Having calculated the aforementioned measures, the method proceeds by sorting tags in the order of decreasing generality and stores them as $\mathcal{V}_{sorted}$. The first element of this list, $\mathcal{V}_{sorted}[0]$, is semantically the most general of all tags and becomes the root tag of the taxonomy. The taxonomy, $\mathcal{T}$, is represented as a set of subsumption axioms between parent and child tags. Formally, each subsumption between parent tag, t_{parent}, and child tag, t_{child}, is represented by $\{t_{parent} \rightarrow t_{child}\}$ such that $\{t_{parent} \rightarrow t_{child}\} \in \mathcal{T}$. The taxonomy is therefore initialized with the root tag as $\mathcal{T} = \{\{\emptyset \rightarrow \mathcal{V}_{sorted}[0]\}\}$ where $\emptyset$ represents a null value, i.e. no parent.

Following initialization, the remaining tags are added to the taxonomy in terms of decreasing generality by calculating the similarity between the tag being added, t_b, and all the tags already in the taxonomy, $\mathcal{T}*$. The tag $t_a \in \mathcal{T}*$ that has the highest similarity with tag t_b becomes the parent of t_b and $\{a \rightarrow b\}$ is added to $\mathcal{T}$. The similarity between tags t_a and t_b, denoted as $\mathrm{S}_{a\rightarrow b}$, measures the degree to which tag t_b is the direct descendant of tag t_a. It is calculated as the degree to which tag t_b is compatible with tag t_a and all the ancestors of t_a:

$$\mathrm{S}_{a\rightarrow b} = \sum_{t_c \in \mathcal{P}_a} \alpha^{l_a - l_c} \frac{\mathrm{D}_{b,c}}{\mathrm{D}_b} \tag{6}$$

Where $\mathcal{P}_a$ is the path in the taxonomy from the root tag $\mathcal{V}_{sorted}[0]$ to tag t_a. l_a and l_c denote the levels in the hierarchy of tags t_a and t_c, respectively. The levels are counted from the root tag starting at zero. Thus, the level of $\mathcal{V}_{sorted}[0]$, denoted as $l_{\mathcal{V}_{sorted}[0]}$, is equal to zero, the levels of its children are equal to one, and so on. The decay factor, α, is a hyperparameter that controls the effect ancestors of tag t_a have on its similarity when calculating $\mathrm{S}_{a\rightarrow b}$. By setting the value of α such that $0 < \alpha < 1$, it's ensured that the effect is lower the more distant an ancestor tag is. The cases were $\alpha = 0$ and $\alpha = 1$ correspond to ancestors having no effect and equal effect on the similarity, respectively.

3.3.2 Hierarchical Clustering Procedure

The induced taxonomy can be used as the foundation of a hierarchical clustering of the knowledge graph's subject entities. Specifically, it is used to initialize the clusters such that each tag in the taxonomy becomes a cluster and the hierarchical relations between tags are extended to the clusters. The tags may then be seen as annotations for each cluster. This is exploited in the notation such that c_a is the cluster initialized from tag t_a. Subject entities are assigned to clusters by the degree to which they belong to a cluster. Belonging of subject s_i to cluster c_a, denoted $\mathrm{B}_{i\rightarrow a}$, is calculated as the Jaccard coefficient between the subject's tags, $\mathcal{A}_i$, and the tags encountered in the path from the root cluster to cluster c_a, denoted $\mathcal{P}_a$. Formally, this is:

$$\mathrm{B}_{i\rightarrow a} = \frac{|\mathcal{A}_i \cap \mathcal{P}_a|}{|\mathcal{A}_i \cup \mathcal{P}_a|} \tag{7}$$

Each subject is added to the cluster to which it has the highest degree of belonging. The process of assigning subjects to clusters may be parallelized to increase performance.

This process may induce a hierarchy containing empty clusters which need to get pruned. Pruning is performed by traversing the hierarchy depth first and removing all empty clusters. In addition, non-empty clusters which have empty parent clusters are reattached as the children of their first non-empty ancestor. If a non-empty cluster has no non-empty ancestors, it becomes the child of the root. The root cluster is never removed, regardless of whether it is empty or not.

3.4 Evaluation

Evaluation of class taxonomy induction methods is difficult as there may be several equally valid taxonomies for a dataset. Previous works such as [44] and [45] have opted for human evaluation, wherein domain experts assess the correctness of relations between classes. [46] used domain experts to rank entire paths on a three point scale. Others, such as [47] and [48], compare class relations against a gold standard taxonomy. In this approach, a confusion matrix between class subsumption axioms is calculated between the induced and gold standard taxonomies. When a gold standard taxonomy can be established, it is the preferred evaluation method as it provides an objective measurement; as such, it is the one that's used in this work. The confusion matrix is used to derive the harmonic mean between precision and recall, the F_1 score [49], as the evaluation metric denoted as Tax-F_1.

The hierarchical clustering is evaluated by calculating the F_1 score of: the belonging of subjects to clusters (Sub-F_1); and how well clusters represent the tags in the vocabulary (Tag-F_1). Sub-F_1 and Tag-F_1 highlight the trade-off between large, heterogeneous clusters on (favoured by Sub-F_1) and smaller, homogeneous clusters (favoured by Tag-F_1). For obtaining the former, each cluster inherits all the subjects of its descendant clusters and the F_1 score is calculated such that a subject is correctly assigned to a cluster if both subject and cluster are annotated by the same tag. The latter is obtained in a way similar to the technique used in [37]. As before, each cluster inherits all the subjects of its descendant clusters and the F_1 score between each tag and each cluster is calculated. The F_1 that is highest among the clusters becomes the score of the tag.

Evaluation was performed on four real-world datasets generated from public online knowledge graphs: Life, DBpedia, WordNet, and IIMB. All four datasets as well as their respective gold standard class taxonomies were generated or acquired during the month of November 2019. The details of these datasets are omitted from this report in interest of brevity and may be found by following the original papers. Each dataset was applied five times to account for the stochasticity in sorting tags of equal generality.

3.4.1 Taxonomy Induction Results

The evaluation results for taxonomy induction are compared with benchmarks in Table 1 and sensitivity to α values is summarized in Figure 3. In general, all tag hierarchy methods achieve encouraging results and the proposed method outperforms the others on two of the four datasets. Notice that since Tax-F_1 measures the balance between precision and recall values, this suggests that the proposed method is both capable of inducing subsumption axioms (recall) while ensuring these axioms are correct (precision). Furthermore, closer inspection of the results reveals that many of the errors can be categorized by two types, which are illustrated by using results from the DBpedia dataset. In the first, the order between parent and child tags are reversed as in the induced {*dbo:Guitarist* → *dbo:Instrumentalist*} when the correct order is {*dbo:Instrumentalist* → *dbo:Guitarist*}. In the second, a tag is misplaced as the child of its sibling. For instance, the gold standard classification of educational institutions is {{*dbo:EducationalInstitution* → *dbo:University*}, {*dbo:EducationalInstitution* → *dbo:College*}} while the induced taxonomy gives the following: {{*dbo:EducationalInstitution* → *dbo:University*}, {*dbo:University* → *dbo:College*}}. Fi-

Table 1: Method results (mean ± standard deviation) on the Life, DBpedia, WordNet, and IIMB datasets.

Method	**Life**	**DBpedia**	**WordNet**	**IIMB**
Heymann and Garcia-Molina	–	0.80 ± 0.02	0.59 ± 0.01	0.20 ± 0.01
Schmitz	0.84 ± 0.00	0.80 ± 0.00	0.79 ± 0.00	0.52 ± 0.00
Paulheim and Fümkranz [50, 39]	–	0.14	–	–
Ristoski et al. [39]	–	0.52	–	–
Proposed method	0.86 ± 0.00	0.88 ± 0.01	0.71 ± 0.01	0.44 ± 0.00

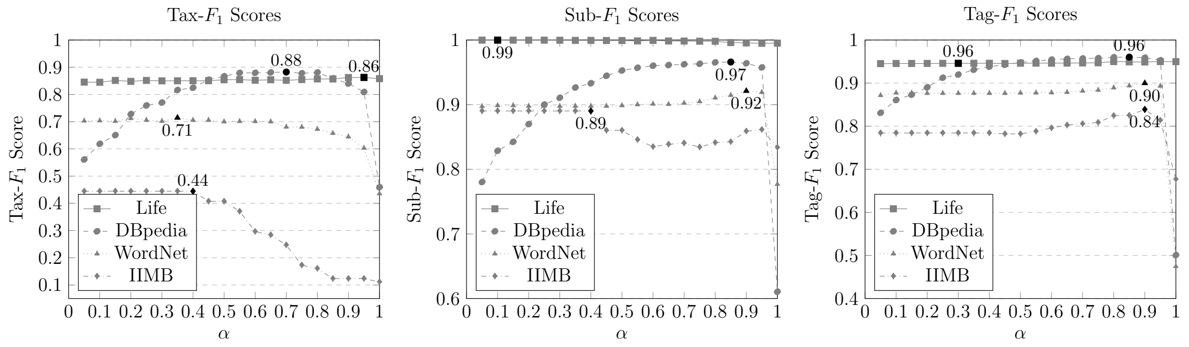

Figure 3: Tax-F_1, Sub-F_1, and Tag-F_1 on the Life, DBpeida, WordNet, and IIMB datasets at various α values.

nally, the induced taxonomy includes subsumption axioms which are considered incorrect as per the gold standard but may not be to a human evaluator. An example of this is that the proposed method induced the subsumption axiom $\{$*dbo:SportFacility* $\rightarrow$ *dbo:Stadium*$\}$ while the gold standard considers $\{$*dbo:Venue* $\rightarrow$ *dbo:Stadium*$\}$ to be the correct parent for *dbo:Stadium*.

3.4.2 Hierarchical Clustering Results

Figure 3 summarizes the Sub-F_1 and Tag-F_1 results of five runs of the hierarchical clustering procedure with different α values. There is no variance in results despite the method's stochasticity in sorting tags. This is because the Sub-F_1 and Tag-F_1 metrics are less sensitive to the ordering errors between parents and children discussed earlier. Furthermore, Sub-F_1 is higher than Tag-F_1 on all datasets. This suggests that the method is better at inducing clusters with a high degree of consistency between cluster members and cluster annotation than it is at representing every class in the taxonomy with a cluster. Closer inspection of clustering errors shows that the majority of errors are the result of errors carried over from the taxonomy induction step. Specifically, the most common type of error is due to missing or incorrect ancestors in the paths of subjects' clusters.

Figure 4 provides an excerpt of hierarchical clustering on the IIMB dataset. Recall that the induced class taxonomy showed a poor Tax-F_1 score on this dataset. Despite this, the hierarchical clustering obtained from this taxonomy scores highly on Sub-F_1 and Tag-F_1 and a qualitative assessment confirms that it is well structured and coherent. This highlights

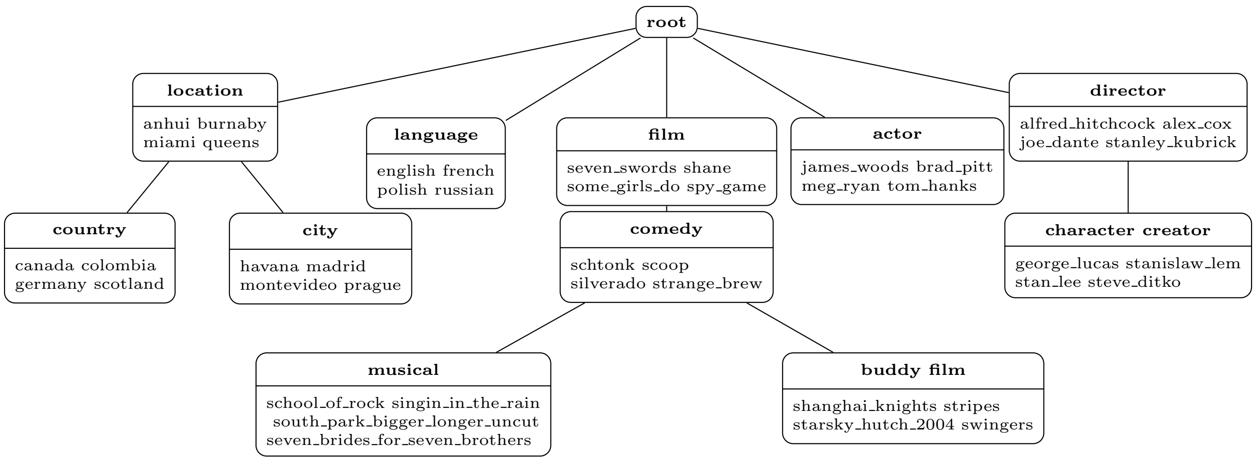
root
location
anhui burnaby miami queens
language
english french polish russian
film
seven_swords shane some_girls_do spy_game
actor
james_woods brad_pitt meg_ryan tom_hanks
director
alfred_hitchcock alex_cox joe_dante stanley_kubrick
country
canada colombia germany scotland
city
havana madrid montevideo prague
comedy
schtonk scoop silverado strange_brew
character creator
george_lucas stanislaw_lem stan_lee steve_ditko
musical
school_of_rock singin_in_the_rain south_park_bigger_longer_uncut seven_brides_for_seven_brothers
buddy film
shanghai_knights stripes starsky_hutch_2004 swingers

Figure 4: Excerpt of the cluster hierarchy induced on the IIMB dataset. Node top indicates cluster's tag; bottom indicates cluster's constituent subjects.

problems with using gold standards, namely: there may be multiple valid ways of structuring a taxonomy; and there may be a disconnect between how the data ought to be structured and how it is structured. Both of these problems are manifest in the IIMB dataset.

4 Work to Date: Probabilistic Coarsening for Knowledge Graph Embeddings

This section summarizes the progress-to-date of the paper *Probabilistic Coarsening for Knowledge Graph Embeddings*. The paper is nearing completion and is intended for publication at an artificial intelligence or semantic web conference.

4.1 Motivation

As previously mentioned, embeddings are among the most common operations on knowledge graphs. Despite this, relatively less work has been done on methods of preprocessing a knowledge graph prior to embedding to yield better results. This provides the motivation for this work which investigates coarsening as a tool for knowledge graph embedding. Specifically, a simple embedding meta-strategy that can be applied to any arbitrary embedding method is proposed. The strategy first reduces an input knowledge graph – henceforth referred to as a base graph – to a coarsened graph along with a mapping between the entities in each knowledge graph. Coarse embeddings are then learned on the coarse graph and mapped back down as base embeddings. They may then be fine tuned on the base graph to reintroduce information that was lost in the coarsening procedure. Figure 5 outlines the flow of this approach.

The rationale for coarsening is multifactorial, as pointed out in [51] and [52]. Specifically:

- Coarsening reduces knowledge graph size whilst preserving global structure, potentially revealing higher-order features.

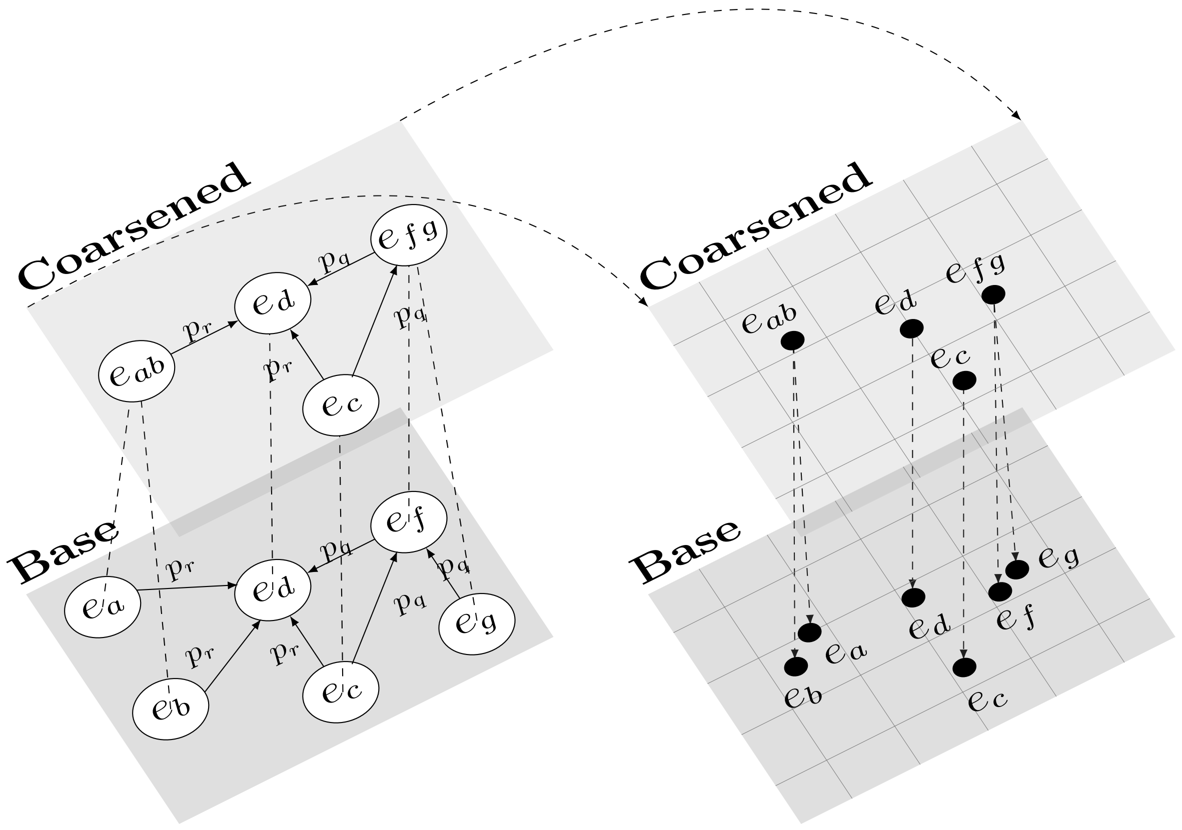

Figure 5: Toy example demonstrating the proposed embedding strategy. The logical flow is guided by dashed line arrows, starting in the bottom left corner and proceeding clockwise.

- Training schemes which rely on stochastic gradient descent may learn embeddings that fall in local minima. Initializations learned on the coarse graph may be more resistant to this problem.
- Structurally equivalent entities are embedded jointly in coarse graphs, reducing training complexity.

4.2 Problem Formulation

Given an arbitrary knowledge graph embedding function, f, the task is to find a mapping which reduces a base graph, $\mathcal{K}$, to a coarse graph, $\mathcal{K}'$, such that $|\mathcal{K}'| < |\mathcal{K}|$. This non-injective surjective mapping, denoted $\Psi : \mathcal{E} \mapsto \mathcal{E}'$ where $\mathcal{E}'$ is the set of entities in the coarse graph, should preserve the global structure of the base graph and be computationally efficient so as to not bottleneck the proposed strategy. Having found Ψ, the proposed strategy can trivially be carried out as outlined earlier.

The tuple structure introduced earlier is extended such that each subject is now identified by both incoming and outgoing relations. Thus, a tag t, is defined as a predicate-entity pair that describes another entity, $t := \langle p_r, e_o \rangle \mid \langle e_s, p_r \rangle$. Order between a tag's predicate and entity corresponds to whether the tag is incoming or outgoing. Thus, each triple in $\mathcal{K}$ corresponds to two entity-tag mappings. As before, these mappings are expressed as sets such that each entity e_a has a corresponding set of tags which annotate it, denoted $\mathcal{A}_a$.

4.3 Approach

The proposed strategy embeds a base graph via an intermediary coarse graph. In this process, the coarse graph is first generated from the base graph before being embedded. Coarse embeddings are then mapped back down to the base graph and fine tuned. The strategy may be divided into the following three steps:

1. **Probabilistic graph coarsening** reduces the base graph to a smaller, coarsened graph and returns an entity mapping between the two graphs.
2. **Coarse graph embedding** applies a predetermined embedding method on the coarse graph to obtain coarse embeddings.
3. **Reverse mapping and fine tuning** maps coarse embeddings back down to the base graph to obtain base embeddings. Base embeddings may be fine tuned on the base graph.

The remainder of this subsection describes each of the these steps in detail.

4.3.1 Probabilistic Graph Coarsening

This procedure involves collapsing structurally similar entities in $\mathcal{K}$ to one entity cluster in $\mathcal{K}'$. Relations in $\mathcal{K}$ are extended to $\mathcal{K}'$ such that a cluster's relations are the union of its constituent entities' relations. Collapsing entities is divided into two stages, designed to preserve the first order and second order proximities of the base graph [12]. This allows the base graph to be reduced of structural redundancies, making it more computationally manageable and potentially revealing its global and most salient features. The mapping between base entities and coarse entities is represented by $\Psi : \mathcal{E} \mapsto \mathcal{E}'$. Coarsening is demonstrated visually on the left half of Figure 5.

Preserving first order proximity refers to the notion that entities should be embedded proximally to their first order (i.e. one-hop) neighbours. By collapsing entities with their first order neighbours, proximity is ensured as collapsed entities share identical embeddings. In undirected, single predicate graphs, edge collapsing [53, 54, 51] finds the largest subset of edges such that no two edges are incident to the same vertex. Vertices incident to each edge in this set are then collapsed, yielding a graph coarsened to preserve first order proximity. Edge collapsing may be applied to knowledge graphs by assuming undirected graph relations. This approach proves too liberal in its coarsening, however, since the cost of coarsening is increased in knowledge graphs due to loss of predicate information. In response, edge collapsing is restricted to entities whose collapsing incurs no loss of predicate information other than predicates which are incident to both entities. Formally, entity e_a is collapsed with e_b if:

$$\{t \in \mathcal{A}_a : e_b \notin t\} \subseteq \{t \in \mathcal{A}_b : e_a \notin t\} \tag{8}$$

First order entity collapsing is demonstrated in the lower left quadrant of Figure 5 where entity e_g is collapsed with its neighbour e_f to form entity cluster e_{fg} in the coarse graph. Note that entities e_a and e_b are also valid candidates for first order collapsing with e_d. This demonstrates the necessity of initially performing second order neighbour collapsing since e_a and e_b are structurally equivalent and thus more similar to one another than to e_d. As such, first order collapsing is performed after second order collapsing.

Second order (i.e. two-hop) neighbours are two entities which share a first order neighbour. The rationale for preserving second order proximity is discussed in [12] and predicated on the intuition that entities which have many common first order neighbours tend to exhibit similar structural and semantic properties. As such, they should be proximal to one another in the embedding space. In Figure 5, notice that second order neighbours e_a and e_b have identical tag sets (i.e. $\mathcal{A}_a = \mathcal{A}_b$) and are thus structurally equivalent. Collapsing these entities and embedding them jointly ensures preservation of second order proximity in the embedding space while incurring no loss of information. This reasoning is applied to the coarsening procedure. Namely, if two second order neighbours exhibit a high degree of structural similarity, they are collapsed in the coarse graph. The similarity between a pair of second order neighbours is measured as the Jaccard coefficient between their tag sets:

$$\text{Sim}(e_a, e_c) = \frac{\mathcal{A}_a \cap \mathcal{A}_c}{\mathcal{A}_a \cup \mathcal{A}_c} \tag{9}$$

Where $\text{Sim}(e_a, e_c)$ is the similarity between e_a and e_c such that $0 \leq \text{Sim}(e_a, e_c) \leq 1$. Second order neighbours are collapsed if their similarity is greater than or equal to a threshold, α, which is chosen such that $0 < \alpha \leq 1$. The value of α dictates the coarseness of the graph with lower α values resulting in smaller, coarser graphs. This process may be seen as a relaxation of Structural Equivalence Matching (SEM) proposed in [52] which collapses second order neighbours only if they are structurally equivalent. In other words, SEM is analogous to the proposed method at $\alpha = 1$; collapsing entities e_a and e_c when $\mathcal{A}_a = \mathcal{A}_c$.

The pairwise comparison between entities and their first and second order neighbourhoods has a worst case time complexity of $O(|\mathcal{E}|^2)$ and is thus computationally infeasible for large scale knowledge graphs. To overcome this, the following scheme is used for sampling neighbours using constrained random walks. To obtain a first order neighbour for entity e_a, first sample a tag from its tag set:

$$t_1 \sim \text{Uniform}(\mathcal{A}_a) \tag{10}$$

Where t_1 represents one hop in a random walk on the base graph. The first order neighbour e_b is then extracted from t_1:

$$e_b = t_1 \cap \mathcal{E} \tag{11}$$

Note that since $t_1 \cap \mathcal{E}$ is a singleton set, the $=$ symbol can be abused such that $e_b = \{e_b\}$. The predicate $p_r = t_1 \cap \mathcal{P}$ is used as a constraint in sampling a second order neighbour for e_a. Specifically, given e_a and its first order neighbour e_b on predicate p_r, only sample second order neighbours which are incident to e_b on p_r:

$$t_2 \sim \text{Uniform}(\{t \in \mathcal{A}_b : p_r \in t \wedge e_a \notin t\}) \tag{12}$$

The second order neighbour e_c is extracted from t_2 analogously to e_b:

$$e_c = t_2 \cap \mathcal{E} \tag{13}$$

Sampled neighbours are collapsed if they meet the aforementioned requirements, resulting in a stochatically derived coarse graph.

For each entity, $\eta \geq 1$ neighbours are sampled resulting in a $O(|\mathcal{E}|\eta)$ time complexity for the sampling scheme. As a hyperparameter of coarsening, η is chosen a priori allowing for flexibility to account for knowledge graph size. In practice it's observed that even small values of η yield encouraging results. The intuition behind this may be summarized as follows:

- Entities which meet the criteria for collapsing are likely to have smaller neighbourhoods.
- Entities that belong to smaller neighbourhoods have a higher probability of getting sampled as candidates for collapsing.

This allows the strategy to be performed with little added computational overhead. Note that reading a dataset has a time complexity of $O(|\mathcal{K}|)$ which may itself be more computationally taxing than coarsening on dense knowledge graphs.

4.3.2 Coarse Graph Embedding

Having coarsened the base graph, coarse embeddings are obtained by applying the arbitrary embedding method f on $\mathcal{K}'$. Since the coarse graph has all the properties of its base counterpart, no additional changes to the embedding method are necessary, merely a different input. Due to there being fewer entities in the coarse graph than its base counterpart, coarse embeddings may require fewer training steps resulting in faster training times. The notation $f(\mathcal{K}')$ is used to denote the embedding of coarse graph $\mathcal{K}'$ to yield coarse embeddings $\mathbf{E}'$:

$$\mathbf{E}' = f(\mathcal{K}') \tag{14}$$

4.3.3 Reverse Mapping and Fine Tuning

Coarse embeddings are extended down as base embeddings, $\mathbf{E}$, by reversing the mapping obtained in the coarsening step:

$$\mathbf{E}[e_a] = \mathbf{E}'[\Psi(e_a)] \tag{15}$$

Where $\mathbf{E}[e_a]$ indexes the base embedding for e_a. A consequence of reverse mapping is that entities which were coarsened together share identical embeddings. In applications which rely on the distinction between these entities, this property is not desired. As such, base embeddings may be fine tuned by embedding $\mathbf{E}$ with respect to the base graph using $\mathbf{E}'$ as initialization. This ensures that structural information which was lost in the coarsening process is reintroduced to base embeddings and collapsed entities become delineated. Furthermore, the training process may be less likely to get stuck in local minima due to its global initializations. The following notation is used to capture fine tuning $\mathbf{E}$ using $\mathbf{E}'$ as initialization:

$$\mathbf{E} = f(\mathcal{K}|\mathbf{E}') \tag{16}$$

Method	**MUTAG**	**AIFB**	**BGS**	**AM**
RDF2Vec	0.7500 ± 0.0392	0.9111 ± 0.0117	0.7828 ± 0.0327	0.8758 ± 0.0143
C(RDF2Vec)	$\underline{\mathbf{0.7956 \pm 0.0340}}$	$\mathbf{0.9167 \pm 0.0000}$	$\underline{\mathbf{0.8828 \pm 0.0178}}$	$\mathbf{0.8778 \pm 0.0211}$
Change	**6.1%***	0.6%	**12.8%***	0.2%
R-GCN	$\mathbf{0.7397 \pm 0.0286}$	0.9528 ± 0.0264	0.8345 ± 0.0424	$\underline{\mathbf{0.8833 \pm 0.0197}}$
C(R-GCN)	0.7294 ± 0.0242	$\underline{\mathbf{0.9694 \pm 0.0088}}$	$\mathbf{0.8690 \pm 0.0317}$	0.8828 ± 0.0138
Change	-1.4%	**1.7%***	**4.1%***	-0.1%
TransE	0.7397 ± 0.0422	0.8722 ± 0.0397	0.6793 ± 0.0371	0.4207 ± 0.0143
C(TransE)	$\mathbf{0.7412 \pm 0.0368}$	$\mathbf{0.9056 \pm 0.0299}$	$\mathbf{0.7759 \pm 0.0335}$	$\mathbf{0.4955 \pm 0.0179}$
Change	0.3%	**3.8%***	**14.2%***	**17.7%***

Table 2: Results of pairwise comparison between the proposed strategy and baseline embedding methods as measured by accuracy (mean ± standard deviation) obtained on testing entities for each dataset. Asterisk (*) indicates superior performance as per Student's t-test at 0.05 level of significance. Underline indicates top performance on dataset, regardless of baseline method.

Dataset	**MUTAG**	**AIFB**	**BGS**	**AM**
Triples	52179	20134	501722	4080981
Change	-29.7%	-30.7%	-45.2%	-31.8%
Entities	16115	2801	78335	944759
Change	-31.8%	-66.2%	-76.5%	-43.3%
Predicates	23	43	97	129
Change	0%	-4.4%	-5.8%	-3.0%

Table 3: Percent reduction in coarse graphs relative to base graphs at $\alpha = 0.5$ and $\eta = 10$.

4.4 Evaluation

The proposed strategy is evaluated on the entity classification task as performed in [20, 21] which involves embedding a knowledge graph and using the embeddings to infer entity labels. The strategy is compared pairwise against the baseline methods used in the embedding step. This allows for measuring whether the coarsening procedure is justified in comparison to using the baseline methods conventionally. Three baseline methods are used to evaluate the proposed strategy: RDF2Vec, R-GCN, and TransE. In performing the evaluation, an implementation was written by extending code provided in [21] and [55]. Four real-world datasets were used in the evaluation: MUTAG, AIFB, BGS, and AM. For each dataset, knowledge graph relations which are on predicates that correlate strongly with its labels were removed as in [21]. A detailed description of each dataset is omitted from this report in interest of space. The notation C(x) is used to refer to the strategy applied to baseline embedding method x. The results of entity classification are summarized in Table 2.

Embeddings were learned using each of the baseline embedding methods on the base graph and on the coarse graph as per the coarsening strategy. To assess the quality of these

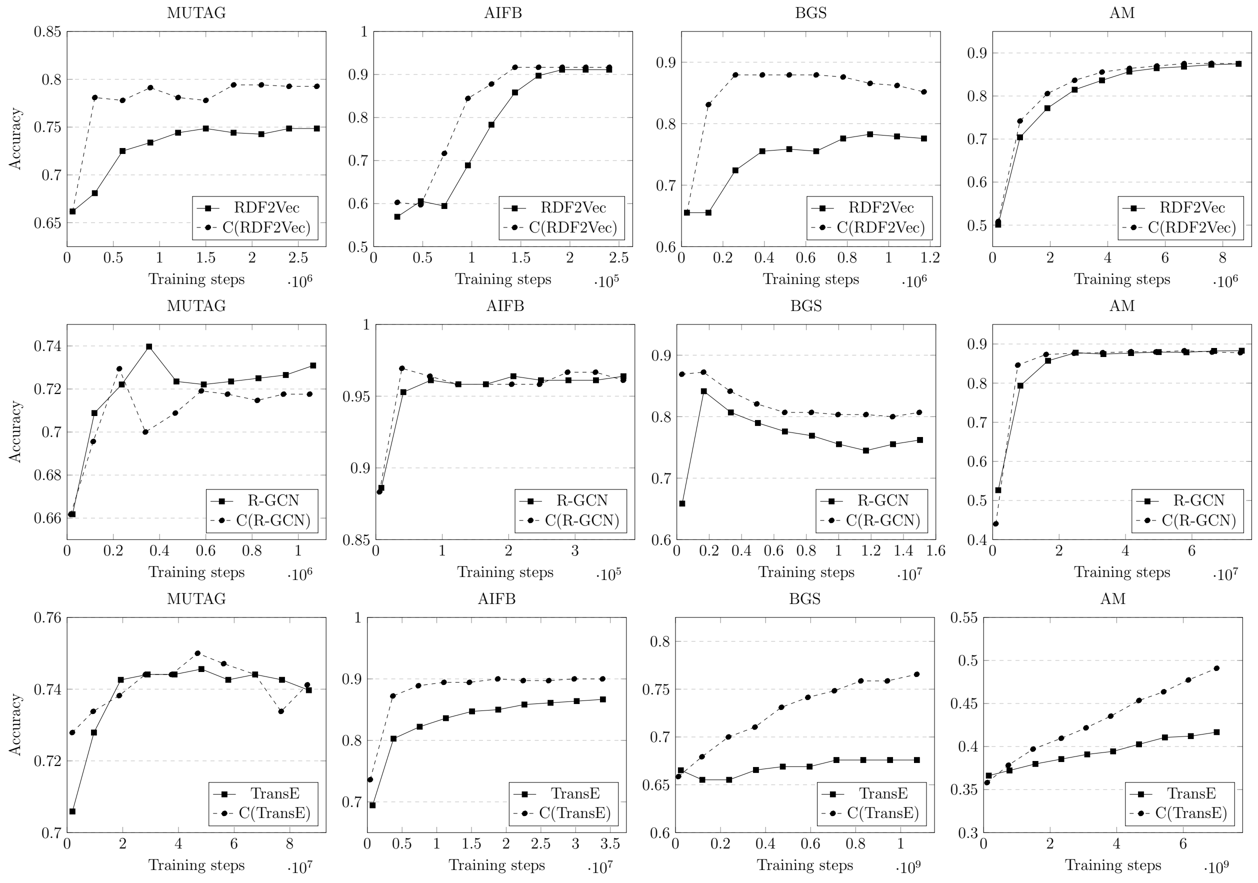

Figure 6: Pairwise comparison between baseline method and the proposed strategy demonstrating performance (accuracy) as a function of the number of training steps performed for each dataset.

embeddings, entity classification was performed by training a support vector machine on 80% on labeled entities and testing on the remaining 20% using splits provided in [20]. The metric for comparing the performance is the accuracy of classification on the testing entities. To account for stochasticity in this process, embeddings were learned and evaluated ten times for each dataset.

To obtain optimal results, 20% of the training entities was set aside as validation for hyperparameter selection and to prevent overfitting. In magnanimity, embedding hyperparameters which were selected on validation results obtained on the base graphs were used for both methods. For coarsening hyperparameters, parameter exploration was performed on $\alpha \in \{0.25, 0.5, 0.75, 1\}$ and used $\eta = 10$ in all of the experiments.

The strategy improved on the baseline in ten of the twelve experiments, seven of which were statistically significant. Furthermore, it was able to achieve state-of-the-art performance on three of the four datasets, albeit using different baseline methods. The strategy appears to perform worse on R-GCN relative to the other baselines. The reason for this may be that coarsening produces graphs with a larger proportion of highly connected hub entities, which is a structural weakness of R-GCN as pointed out by its authors. Finally, datasets with knowledge graphs that have a higher degree of reduction at $\alpha = 0.5$ and $\eta = 10$

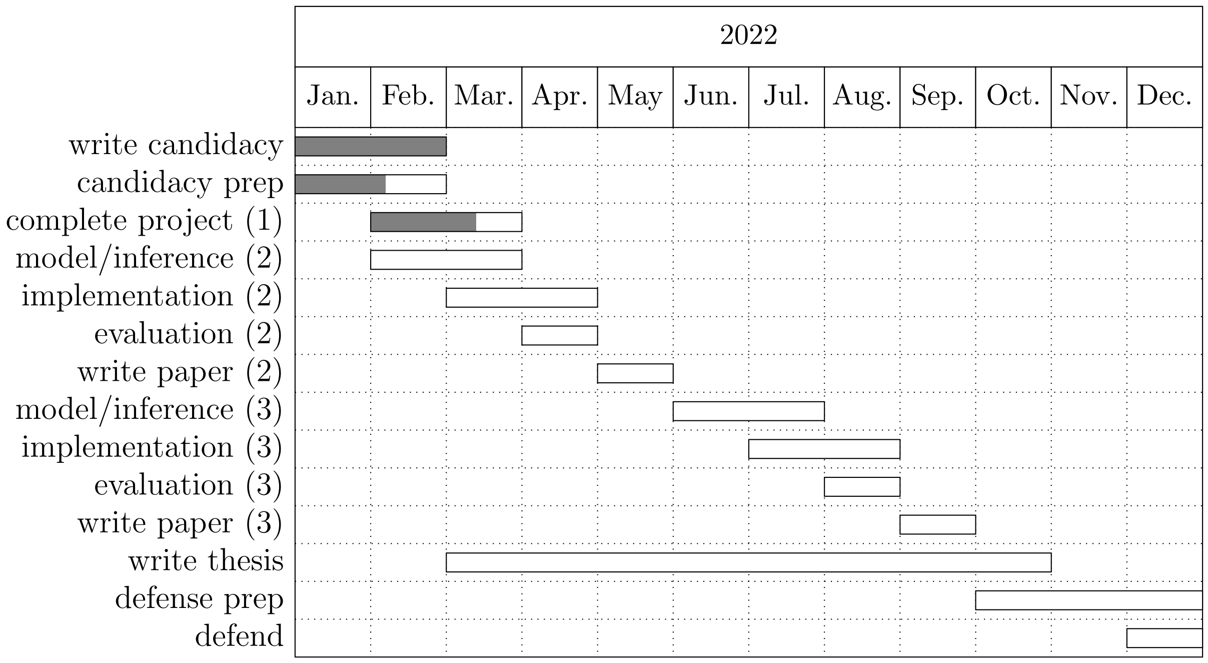

Figure 7: Gantt chart of the anticipated work plan leading up to the final defense. Numbers in parenthesis correspond to the three projects described in the Future Work section.

perform better. This is because not all knowledge graphs are equally suitable candidates for coarsening. Namely, knowledge graphs which exhibit a high degree of structural equivalency between entities lose less information in the coarsening process. Notice this in the AIFB and BGS datasets where more than half of their entities get collapsed in the coarsening step as shown in Table 3.

Figure 6 plots the performance of the proposed strategy compared to baselines when increasing the number of training steps performed. Due to computational constraints, the embeddings were trained up to fifty epochs. It is possible that given enough training, baseline methods could catch up in performance to the coarsening strategy as can be seen on the AM dataset using TransE. This, however, still demonstrates that coarse graphs produce quality embeddings faster than embedding on base graphs. This is further confirmed by RDF2Vec on AIFB and AM and R-GCN on MUTAG which show similar trendlines with the coarsened counterpart requiring fewer training steps. This suggests that the coarsening strategy is faster to train than its baseline counterparts.

5 Future Work

This section briefly outlines the work expected to be completed before the final defense. The monthly breakdown of this work is summarized in the Gantt chart in Figure 7.

5.1 Further Investigate Graph Coarsening for Embeddings

The strategy proposed in the previous section, although fully implemented and evaluated in its current formulation, deserves further investigation. Specifically, there are two avenues for development. First, the proposed strategy is to coarsen the graph once before embedding. An alternative to this is to coarsen the graph repeatedly, generating a hierarchy of progressively coarsened graphs. Formally, for a predetermined number of coarsening steps, K, generate K coarsened graphs, denoted $\mathcal{K}^1, \mathcal{K}^2, ..., \mathcal{K}^K$. Then, embed the coarsened knowledge graphs iteratively starting with the coarsest graph and proceeding downwards in the hierarchy using the embeddings learned in the preceding level as the initialization. The general rule for embedding the knowledge graph at level l is then:

$$\mathbf{E}^l = f(\mathcal{K}^l | \mathbf{E}^{l-1}) \tag{17}$$

The second avenue is to reevaluate the coarsening procedure. Specifically the collapsing of first order neighbours loses predicate information which may indicate that collapsed entities should not be embedded jointly. To illustrate this, consider the triple $\langle dbr : Cranmore, owl : differentFrom, dbr : Carnmore\rangle$. Collapsing these first order neighbours and embedding them jointly would not be correct because $dbr : Cranmore$ and $dbr : Carnmore$ are not sufficiently similar semantically as evidenced by the linking predicate $owl : differentFrom$. In practice, this problem occurs rarely since rarely would two such entities meet the condition in Equation 8.

5.2 Hierarchical Blockmodelling for Knowledge Graphs

Recall that stochastic blockmodels decompose a graph by assuming an underlying structure and formulating it in the form of probability distributions. In assuming a fundamentally hierarchical structure to a knowledge graph, it is possible to learn an explicit hierarchy through the model's generative process. The key to doing this is to find a statistical framework to learn a hierarchy non-parametrically and intergrate it with the model that makes solving the inference possible[5].

The Nested Chinese Restaurant process (nCRP) [56] is a stochastic process that has been used as an infinitely deep and infinitely branching prior over a tree structure. Although its specifics are outside the scope of this report, at a high level the nCRP assigns a probability distribution for travelling down and creating new paths in a tree. Perhaps most famous for its use in topic modelling, it has also been used in stochastic blockmodelling as a statistical framework for hierarchically organizing groups of nodes [57]. Put simply, it's possible to use the nCRP to generate a hierarchy of entities from a knowledge graph. A problem inherent to stochastic blockmodels is that of scalability. This is due to the modelling of all pairwise interactions between nodes in the graph. This issue can be surmounted by, for instance, marginalizing over model parameters – integrating them out of the learning process – in

[5] *Remark: Although omitted from this candidacy report, I have prior experience in developing stochastic blockmodels for graph generation including authoring one and co-authoring three conference papers in this field. I anticipate, therefore, that devising the model, implementation, and evaluation will be relatively straightforward. Developing the inference procedure for the posterior distribution will be less straightforward as I have no prior experience solving the necessary inference schemes such as Gibbs sampling.*

what is known as collapsing a Gibbs sampler. This process is inextricably linked to solving the inference and presents the major challenge moving forward in this work.

5.3 Knowledge Graph Abstraction

Stochastic blockmodels may be utilized as a tool for knowledge graph abstraction since, at their core, they are clustering methods which probabilistically model the relationships between clusters. For instance, consider a knowledge graph that includes countries and cities related by the predicate $dbo : capital$. A stochastic blockmodel may learn the clusters for countries and cities as well as a probability that entities in the country cluster relate to the entities in the cities cluster. The clusters along with their relations to other clusters may be thought of as an abstract version of the underlying knowledge graph. Although this project is still in its infancy and exists merely as a collection of loosely joined ideas, it is anticipated to be realized before the defense and is thus mentioned in this report.

References

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[2] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "Yago2: A spatially and temporally enhanced knowledge base from wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.

[3] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledge base," 2014.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, AcM, 2008.

[5] A. Singhal, "Introducing the knowledge graph: things, not strings." https://www.blog.google/products/search/introducing-knowledge-graph-things-not/, 2012, accessed January 28, 2022.

[6] R. Xie, Z. Liu, M. Sun, *et al.*, "Representation learning of knowledge graphs with hierarchical types.," in *IJCAI*, vol. 2016, pp. 2965–2971, 2016.

[7] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3065–3072, 2020.

[8] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of sparql," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, pp. 1–45, 2009.

[9] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An evolving query language for property graphs," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 1433–1445, 2018.

[10] D. Archdeacon, "Topological graph theory," *A survey. Congressus Numerantium*, vol. 115(5-54):18, 1996.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015.

[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[15] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, pp. 2224–2232, 2015.

[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[18] V. Bellini, A. Schiavone, T. Di Noia, A. Ragone, and E. Di Sciascio, "Knowledge-aware autoencoders for explainable recommender systems," in *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, 2018.

[19] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International Conference on Artificial Neural Networks*, pp. 412–422, Springer, 2018.

[20] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *International Semantic Web Conference*, pp. 498–514, Springer, 2016.

[21] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, pp. 593–607, Springer, 2018.

[22] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," *arXiv preprint arXiv:1707.01476*, 2017.

[23] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, pp. 2787–2795, 2013.

[24] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data.," 2011.

[25] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[26] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *arXiv preprint arXiv:1711.05851*, 2017.

[27] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition and applications," *arXiv preprint arXiv:2002.00388*, 2020.

[28] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American statistical association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[29] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," in *AAAI*, vol. 3, p. 5, 2006.

[30] E. M. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed membership stochastic blockmodels," *Advances in neural information processing systems*, vol. 21, 2008.

[31] Q. Han, K. Xu, and E. Airoldi, "Consistent estimation of dynamic and multi-layer block models," in *International Conference on Machine Learning*, pp. 1511–1520, PMLR, 2015.

[32] S. Paul and Y. Chen, "Consistent community detection in multi-relational data through restricted multi-layer stochastic blockmodel," *Electronic Journal of Statistics*, vol. 10, no. 2, pp. 3807–3870, 2016.

[33] P. Barbillon, S. Donnet, E. Lazega, and A. Bar-Hen, "Stochastic block models for multiplex networks: an application to a multilevel network of researchers," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 180, no. 1, pp. 295–314, 2017.

[34] A. R. Pamfil, S. D. Howison, and M. A. Porter, "Edge correlations in multilayer networks," *arXiv preprint arXiv:1908.03875*, 2019.

[35] C. De Bacco, E. A. Power, D. B. Larremore, and C. Moore, "Community detection, link prediction, and layer interdependence in multilayer networks," *Physical Review E*, vol. 95, no. 4, p. 042317, 2017.

[36] J. Völker and M. Niepert, "Statistical schema induction," in *Extended Semantic Web Conference*, pp. 124–138, Springer, 2011.

[37] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing yago: scalable machine learning for linked data," in *Proceedings of the 21st international conference on World Wide Web*, pp. 271–280, ACM, 2012.

[38] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, pp. 49–60, ACM, 1999.

[39] P. Ristoski, S. Faralli, S. P. Ponzetto, and H. Paulheim, "Large-scale taxonomy induction using entity and word embeddings," in *Proceedings of the International Conference on Web Intelligence*, pp. 81–87, ACM, 2017.

[40] M. Pietrasik and M. Reformat, "A simple method for inducing class taxonomies in knowledge graphs," in *European Semantic Web Conference*, pp. 53–68, Springer, 2020.

[41] M. Pietrasik and M. Reformat, "Path based hierarchical clustering on knowledge graphs," *arXiv preprint arXiv:2109.13178*, 2021.

[42] P. Schmitz, "Inducing ontology from flickr tags," in *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, vol. 50, p. 39, 2006.

[43] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," tech. rep., 2006.

[44] C. Gu, G. Yin, T. Wang, C. Yang, and H. Wang, "A supervised approach for tag hierarchy construction in open source communities," in *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, pp. 148–152, ACM, 2015.

[45] W. Wang, P. M. Barnaghi, and A. Bargiela, "Probabilistic topic models for learning terminological ontologies," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 1028–1040, 2009.

[46] S. Wang, D. Lo, and L. Jiang, "Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pp. 604–607, IEEE, 2012.

[47] K. Liu, B. Fang, and W. Zhang, "Ontology emergence from folksonomies," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1109–1118, ACM, 2010.

[48] F. Almoqhim, D. E. Millard, and N. Shadbolt, "Improving on popularity as a proxy for generality when building tag hierarchies from folksonomies," in *International Conference on Social Informatics*, pp. 95–111, Springer, 2014.

[49] N. Chinchor, "Muc-4 evaluation metrics," in *Proceedings of the 4th conference on Message understanding*, pp. 22–29, Association for Computational Linguistics, 1992.

[50] H. Paulheim and J. Fümkranz, "Unsupervised generation of data mining features from linked open data," in *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, p. 31, ACM, 2012.

[51] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *Proc. 32nd AAAI Conf.Artif. Intell.*, pp. 2127—2134, 2018.

[52] J. Liang, S. Gurukar, and S. Parthasarathy, "Mile: A multi-level framework for scalable graph embedding," *arXiv preprint arXiv:1802.09612*, 2018.

[53] B. Hendrickson and R. W. Leland, "A multi-level algorithm for partitioning graphs.," *SC*, vol. 95, no. 28, pp. 1–14, 1995.

[54] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[55] X. Han, S. Cao, L. Xin, Y. Lin, Z. Liu, M. Sun, and J. Li, "Openke: An open toolkit for knowledge embedding," in *Proceedings of EMNLP*, 2018.

[56] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *Journal of the ACM (JACM)*, vol. 57, no. 2, pp. 1–30, 2010.

[57] Q. Ho, A. Parikh, L. Song, and E. Xing, "Multiscale community blockmodel for network exploration," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 333–341, JMLR Workshop and Conference Proceedings, 2011.